



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/769,535	01/30/2004	Milton E. Moskowitz	H0005134- 1623	8642
128	7590	06/18/2008		
HONEYWELL INTERNATIONAL INC. 101 COLUMBIA ROAD P O BOX 2245 MORRISTOWN, NJ 07962-2245			EXAMINER	
			VU, TUAN A	
			ART UNIT	PAPER NUMBER
			2193	
			MAIL DATE	DELIVERY MODE
			06/18/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No. 10/769,535	Applicant(s) MOSKOWITZ ET AL.
	Examiner Tuan A. Vu	Art Unit 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
 - If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
 - Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED. (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) Responsive to communication(s) filed on 17 March 2008.
 2a) This action is FINAL. 2b) This action is non-final.
 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) Claim(s) 1-18,21-22 is/are pending in the application.
 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
 5) Claim(s) _____ is/are allowed.
 6) Claim(s) 1-18,21 and 22 is/are rejected.
 7) Claim(s) _____ is/are objected to.
 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) The specification is objected to by the Examiner.
 10) The drawing(s) filed on 1/30/04 is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date: _____ |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1668) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 3/17/08.

As indicated in Applicant's response, claims 1-18, 21-22 have been amended. Claims 1-18, 21-22 are pending in the office action.

Specifications: Drawings

2. The Drawings is objected to because of the inconsistencies of legend in Figure 2 with respect to the section of the Disclosure depicting this Figure; the mismatch between the numerals of the Drawings and their counterparts described in para [0027] pg. 6 to pg. 7 of the Specifications are numerous. Examples from the drawings are elements code_file 101, VOB 112, display 204 which do not have the proper matchings in the above paragraph. Throughout the Specifications, database 212 is mentioned not VOB 112. Correction is required.

Claims Objections

3. Claims 1, 7, 12 are objected to because of the following impropriety: the reciting of 'expected computer code' from the model file (i.e. "processing the model file to generate an expected code"; "generate an expected computer code from the model") when compared with the 'generated computer code' being formed from the same model file cannot justify the use of 'expected' qualifier because, for example claim 1, there is nothing in the language about generating "from a model file" that would make the second plurality of lines of code to possibly be 'expected code' while the first plurality is merely 'generated' code. One would not be convinced that by using a same model file, the computer code generated the first time would be any different code generated from that file a second time, thus the process as to determining error from comparing 'expected' code against actual ('generated') code as claimed is not reasonably

accepted. The ‘expected’ terminology for not having further been specified with additional teachings that would render its use appropriate, is deemed a far-fetched terminology and should be corrected, simply because processing a same model file to derive a computer code, there is no distinguishing details as to why a second code generation would yield a ‘expected code’ whereas, using that same model, a first code generation would not. The terminology is deemed unreasonable and should be corrected to merely depict that the generated code is first copy and that the so-recited ‘expected code’ is but a second copy thereof, unless the claim be amended to include further teachings that otherwise convey how ‘expected’ code is more compliant to the first ‘generated’ code. The ‘expected code’ will be treated as mere code form, or syntax, expressions defined by language compliancy rules as observed in a compilation process or code auditing process.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 1-18, 21-22 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, claim 1 recites ‘processing the model file to *generate* an expected computer code having a second plurality of lines from the model file, each line in the second plurality of

lines corresponding to a line in the first plurality of lines'. There is insufficient teaching in the Disclosure as to enable one to perceive that processing a model file, the Verification module 102 (Figure 1) does generate a second plurality of lines from the model file, each lines thereof corresponding to a line in the first plurality of lines. According to the Disclosure regarding storage 210 of Figure 2, results from the verification (see Specifications: top pg. 7) of the generated code is in output file 108, and this does not describe any generation of lines of code per se from a model file by this module 102. Further, shorthand notations of generated code 320 (Figure 3b) are replaced to enable comparisons with header information when verification module 102 executes (see para 0032, pg. 9), with the expected header information being stored in database 212; i.e. no generation of header information lines by this verification module 102.

Regarding the flow of Figure 4, step 412 only depicts if expected value of 3.14159 is matched with the corresponding block in code 320; i.e. no generation per se is depicted to yield any second plurality of lines by verification module 102. Further, step 414 of Fig. 4 describes a matching process such that string comparison between code 320 is made (see para 0036) against expected *form or command* stored in database 212, i.e. the pre-stored expected form or command here not same as actually generating of plurality of code lines by the verification module in the course of processing *model_file* 101. Likewise, determination by module 102 regarding expected lines of code from a summation block being (Specs, para 0040) remains a check whether a **proper syntax should be** effectuated with respect to the actual line generated, hence, no (plurality of lines) generation per se depicted here. Step 418 depicts whether expected lines of code are present (para 0046, pg. 11) when matched against code 320 to see if empty spaces should be resulting; no plurality of lines generated per se by module 102 in processing a model

file (emphasis added) and while comparing against code 320; and likewise, in step 420 a determination is made to verify whether all expected code lines are generated. As a whole, from the description of Figure 4, determination (by module 102) as to whether code lines (from Fig. 3b) have been generated or contain proper syntax form, is not same as *actually regenerating* (a second copy of) code lines 320 from the exact same model file 101 (see Figure 1), as recited from the above claim language ("processing the model file ... to generate ... a second plurality of lines ... from the model file"). The lack of equivalent from the Disclosure in regard to the 'generate' limitation is perceived as a lack of description, as though the inventor has no possession of this limitation, and one would not be sufficiently taught in order to make use of the invention as claimed. This improperly supported limitation would be treated as a mere dynamic check or determination by a code verification process as to what exact syntax or expected code form (e.g. being preordained in a database) the generated lines (e.g. code 320) should be comply to; that is, no patentable weight given to the action of 'to generate ... plurality of lines from the model'.

Claims 7, 12 recite 'generate an expected code ... from the model file, each line of the expected code ... corresponding line of generated code'; and are rejected for the same lack of Disclosure description.

Claims 2-6, 8-11, 13-18, 21-22 fail to remedy to the lack of enabling description from the Disclosure, and are rejected likewise.

6. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claims 1-18, 21-22 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Specifically, claims 1, 7, 12 are rejected as being incomplete for omitting essential elements, such omission amounting to a gap between the elements. See MPEP § 2172.01. The omitted elements are: what is supporting the generating of 'expected computer code' that would make it 'expected' rather than merely 'generated computer code' when the 'generating' of both 'generated computer code' and 'expected computer code' appears to be identical. The claim language regarding 'expected computer code' being generated from the same model file as the 'generated code' has been deemed not definite in the fact that since the 'expected' code is to validate the correctness of the 'generated code', the 'expected computer code' should be obtained with a different manner, using more compliant rules. As set forth in the Claim Objections to the improper use of 'expected', there is no sufficient teaching in the way the 'expected computer code' is generated for it to be any different than the 'generated computer code'. According to the Specifications, a parsing process (see para 0033, 0036, pg. 9, 10) operates on a model or operates based upon pre-stored command or syntactic form from a database. Because the claim language as set forth above is devoid of specifics to differentiate the nature of 'expected' code versus 'generated' code, the claim is indefinite in establishing the fact that 'expected' code is such as to serve as basis to validate the 'generated' code. One of ordinary skill in the art would not be able to make use of the invention just by generating a first code based on a model file, then process to generate another code from that same model file, and validate the correctness of the former code ("generated code") with the latter ("expected code"). There is a lack of utility or essential steps teaching (emphasis added) that is deemed instrumental

in making the second instance of generated code ‘expected’ status (as opposed to merely ‘generated’ status of the first instance of generated code). For the sake of expediting prosecution of the claims, the ‘expected code’ will be treated as mere language compliant constructs, form, expression, syntax, code strings derived by a validation engine and utilized as basis to support the language compliancy of the ‘generated code’.

Claims 2-6, 8-11, 13-18, 21-22 are rejected for not remedying to the above indefinite language deficiency.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

9. Claims 1-18, 21-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Charisius et al, USPN: 6,983,446.

As per claim 1, Charisius discloses a method for verifying a generated computer code having a first plurality of lines (e.g. Fig. 4; 812, Fig. 8C; 1302, Fig. 13; 1912, Fig. 19C) generated from a model file of a system comprising:

processing the model file (Fig. 3; Fig. 13-14) to generate (Note: *generate* treated as *determine* – see USC § 112, 1st para Rejection) an expected computer code (e.g. Steps 2000-2004, 2008, Fig. 20A, Table 1 – Table 18, col. 10-20 – Note: Processing TMM model then invoke a verification tool **reads on** processing to determine code compliancy, and metrics based on which to enforce language compliancy in terms of programming semantic and syntactic errors

or audit violation – Basic Metrics, Cohesion Metrics, Complexity Metrics, inheritance Metrics, maximum metrics, Ratio Metrics, Style audits, Declaration audits, Name style, Superfluous content; that is, expected code having syntax or form being language compliant when verified against the first plurality of lines that have been generated);

comparing each line in the first plurality of lines to determine if the generated computer code and the expected computer code match (e.g. Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13 – Note: actual source code using the SCI reads on *generated code* lines 810, 812, 1302; and audit module to validate expected code being model specifications that encompass rules definitions – audited by metrics - reads on comparing expected and generated; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A – reads on expected form and syntax being determined in TMM form via template of lines – col. 16, lines 9-14); and

transmitting an error message (e.g. col. 3, lines 38-44, 55-63) if the generated computer code and the expected computer code do not match.

But Charisius does not explicitly disclose expected computer code having a second plurality of lines, each line in the second plurality of lines corresponding to a line in the first plurality of lines; nor does Charisius explicitly disclose comparing each line in first plurality of lines and each corresponding line in said second plurality of lines. Based on the checking for form, syntax and language rule semantic being formulated by metrics (e.g. Table 1 – Table 18, col. 10-20) and processing TMM model by Charisius using a verification tool to determine code compliancy (Fig. 19A, 20A) in light of metrics based on which to enforce language compliancy in terms of programming semantic and syntactic errors or audit violation (see: Basic Metrics,

Cohesion Metrics, Complexity Metrics, inheritance Metrics, maximum metrics, Ratio Metrics, Style audits, Declaration audits, Name style, Superfluous content, col. 10-20), verification of actual code form being generated for compliancy with respect to expected code syntax or form entails that a plurality of compliant syntax constructs or form are to be determined when performing the verification (against metrics) or audit process by Charisius, and this can be seen in a replacement of source lines by a more appropriate one (see Charisius: *may be replaced by* - col. 23-24). It would have been obvious for one skill in the art at the time the invention was made to implement Charisius's verification or auditing tool --using the above audits rule or metrics -- so that a plurality of code forms or constructs predefined as compliant, reference or expected would be dynamically determined (i.e. expected code having a plurality of lines, each of which corresponding to a line of the first plurality) when performing code style or syntax compliancy check as set forth above. One would be motivated to do so because by having determined such plurality of expected code forms verifying compliancy of the first plurality of code lines against said expected or reference plurality of forms or syntaxes, would enable an extensive line per line or form per form comparison, so to yield discrepancy between each and every actual code form/line being generated with respect to each and all corresponding reference/expected code form/line as purported by a verification or auditing tool, to apply a proper corrective action (e.g. col. 23-24, 26-27)

As per claims 2-3, Charisius (based on the comparing of each expected code line with each generated code line – as set forth in claim 1) discloses steps of:

comparing each line in the first plurality of lines and each corresponding line in the second plurality of lines to determine if the first plurality of lines does not include a line of code

included in the second plurality of lines (e.g. *synchronized ... updated automatically* – col. 5, lines 34-60; Fig. 20B; *Counts the number of code lines* -Table 1; col. 41 lines 40-62 – Note: determine as whether a corrected source code should include some construct reads on first plurality not included expected construct of second plurality) or if the first plurality of lines includes any line of code not in the second plurality of lines (see col 42, lines 45-65).

But Charisius does not explicitly teach transmitting an error message if the first plurality of lines under analysis does not include a line of code included in the second plurality of lines, or if the first plurality of lines includes any line of code not in the second plurality of lines. But based on the message error to let the developer be visually informed from the onset (e.g. col. 3, lines 38-44, 55-63; *wrong* 810 – Fig. 8B), it would have been obvious for one skill in the art at the time the invention was made to implement the source code auditing by Charisius so that when any code form, line count, construct logical order verified in terms of metric or auditing requirement as set forth above does not match with the that of expected code, some error messages would be visually generated in order for the developer to effectuate proper verification of the intended target code.

As per claim 4, Charisius (based on the comparing of expected code line with the generated code line – as set forth in claim 1) discloses:

comparing each line in the first plurality of lines and each corresponding line the second plurality of lines to determine if the first plurality of lines are in a similar logical order as the second plurality of lines (see Table 1, Table 2, table 3, table 4, col. 10-12; Fig. 19b, 19C; col. 4 line 66 to col. 5, line 9); but Charisius does not explicitly disclose transmitting the error message

if the first plurality of lines are not in the similar logical order; but this has would have been obvious for the same reataionale as set forth above.

As per claims 5-6, Charisius (based on the comparing of expected code line with the generated code line – as set forth in claim 1) discloses

comparing a first header information section in the first plurality of lines and an expected a second header information section in the second plurality of lines to determine if the first header information section matches the second header information section; comparing a first declared variable section in the generated first plurality of lines to a second declared variable section in the second plurality of lines to determine if the first declared variable section matches the second declared variable section (*Declaration* – cols. 25-26; match a declaration, col. 37, lines 1-37- Note: generated source code or class package declaration with respect to expected declaration in OO class or Use case package – see Fig. 14-15, 22 -- in an *audit* instance reads on comparing header of a class signature declaration).

But Charisius does not explicitly teach transmitting the error message if the first header information section does not match the-second header information section; or if the first declared variables variable section does not match the second declared variable section. But this error transmitting limitation has been addressed in claims 2-4 above.

As per claim 7, Charisius discloses a computer-readable storage medium containing a set of instructions for verifying a generated computer code having a first plurality of lines, the generated computer code automatically generated from a model file of a system (Fig. 4; 812, Fig. 8C; 1302, Fig. 13; 1912, Fig. 19C), the set of instructions comprising code that:

reads in the model file (Fig. 3; col. 5, lines 50-64; *Use case Modeling* - col. 15, lines 37-44, 55-64);

generates (Note: *generate* treated as *determine* – see USC § 112, 1st para Rejection) an expected computer code (e.g. Steps 2000-2004, 2008, Fig. 20A, Table 1 – Table 18, col. 10-20 – Note: Processing TMM model then invoke a verification tool **reads on** processing to determine code compliancy, and metrics based on which to enforce language compliancy in terms of programming semantic and syntactic errors or audit violation – Basic Metrics, Cohesion Metrics, Complexity Metrics, inheritance Metrics, maximum metrics, Ratio Metrics, Style audits, Declaration audits, Name style, Superfluous content; that is, expected code having syntax or form being language compliant when verified against the first plurality of lines that have been generated)

reads in the generated computer code; compares each line in the first plurality of lines and the expected code to determine if the generated computer code and the expected computer code match (Fig. 4; Fig. 19B-C; Fig. 8B-C; code 1302 – Fig. 13 – Note: actual source code using the SCI reads on *generated code* lines 810, 812, 1302; and audit module to validate expected code being model specifications that encompass rules definitions – audited by metrics - reads on comparing expected and generated; Fig. 3; Fig. 13-14; col. 5, lines 61-64; col. 6, lines 12-22; Note: *definitions, templates*, Fig. 8B-C; col. 7, lines 63 to col. 8, line 21; Fig. 19A, 20A – reads on expected form and syntax being determined in TMM form via template of lines – col. 16, lines 9-14); and

transmits an error message if the generated computer code and the expected computer code do not match (e.g. col. 3, lines 38-44, 55-63).

But Charisius does not explicitly disclose expected computer code having a second plurality of lines, each line in the second plurality of lines corresponding to a line in the first plurality of lines; nor does Charisius explicitly disclose code that compares each line in first plurality of lines and each corresponding line in said second plurality of lines. But this limitation has been rendered obvious in claim 1.

As per claims 8-10, refer to claims 2-4, respectively.

As per claim 11, Charisius discloses code that compares a first header information section (Note: signature of a OO Class reads on a formal header declaration – see cols. 25-26) in the first plurality of lines and a second header information section in the second plurality of lines to determine if the first header information section matches the second header information section (refer to claim 5).

But Charisius does not explicitly teach transmitting an error message if the header section of generated code under analysis does not match the expected header, but this error transmitting limitation has been addressed in claims 2-4 above. (refer to rationale set forth in claim 5).

As per claim 12, Charisius discloses a system for verifying the contents of a generated computer code generated from a model file, comprising a processor operable to:

generate an expected computer code from the model file;
compare each line in the generated computer code with the expected computer code, and
transmit an error message if the generated computer code and the expected computer code do not match; and

a display configured to display the error message, the display coupled to the processor;
all of which having been addressed in claim 7.

But Charisius does not explicitly disclose expected computer code with each line in the second plurality of lines corresponding to a line in the first plurality of lines; nor does Charisius explicitly disclose code that comparing each line in generated code and each corresponding line in said expected code; and generating error message if each line in the generated code and each line in the expected code do not match.

But the expected code having lines each of which corresponds to one in the generated code so that the comparing is effectuated upon each line of expected and generated code (including the error message based upon comparing) fall under the rationale being set forth in claim 1; i.e. these limitations would have been obvious by virtue of claim 1.

As per claims 13-14, Charisius (based on the rationale of claim 1) discloses wherein the results of the comparison indicates if each line of the generated computer code has all of the content of each corresponding line in the expected computer code (e.g. Fig. 8A-B; Fig. 20; *synchronized ... updated automatically* – col. 5, lines 34-60); wherein the results of the comparison indicates if the generated computer code has any additional content not found in the expected computer code (e.g. col. 5, lines 34-60– Note: auditing tool to match each constructs of the lines of code with format required for OO syntax construction based on template and graphical representation – see Figs 11, 19, 20 – maps with indication as to any additional content is not found – see *update view* Fig. 9; *incremental code editor* – Fig. 7).

But Charisius does not explicitly teach transmitting an error message if in the generated code all the content is not matched; or if any additional content is not found. However, this error transmitting limitation has been addressed in claims 2-4 above.

As per claims 15-18, refer to claims 2-5, respectively.

As per claims 21-22, Charisius (based on the comparing of each expected code line with each generated code line – as set forth in claim 1) discloses comparing each line of the generated computer code (e.g. Fig. 4; Fig. 19B-C) to each line of the expected computer code to determine if the generated computer code includes all of the lines (e.g. *synchronized ... updated automatically* – col. 5, lines 34-60) of the expected computer code.

But Charisius does not explicitly disclose generating an error if the generated code does not include all the lines of the expected code. However, this error transmitting limitation has been addressed in claims 2-4 above.

Response to Arguments

10. Applicant's arguments filed 3/17/08 have been fully considered but they are mostly moot or not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 102 Rejection:

- (A) Applicants have submitted that Charisius cannot teach that 'each line in the second plurality of lines ... corresponding to a line in the first plurality of lines' (Appl. Rmrks, pg. 10). The arguments correspond to a newly added limitations, hence is deemed moot.
- (B) Applicants further have submitted that as amended, claim 1 recites 'each line in the generated code is compared to a corresponding line of expected code'; and that Charisius only teaches generating one source code (Appl. Rmrks pg. 11). The new grounds of rejection along with the USC § 112 of rejection have been necessitated by the amendments; i.e. the above argument being moot.
- (C) Applicants have submitted that UML model class definitions used in Charisius is for syntax compliance according to a compiler process, which is not comparing each line of

Art Unit: 2193

expected code being generated with each corresponding line of generated code as claimed (Appl. Rmrks pg. 12). The argument is moot in view of the new grounds of rejection; and the ‘generating’ of expected code lines has been addressed as a USC 112 deficiency.

USC § 103 Rejection:

(D) Applicants have submitted that because Charisius deficiency in fulfilling the base claims, the subject matter of dependent claims 2-6, 8-11, 13-18, 21-22 would not have been obvious (Appl. Rmrks pg. 14). The rejection herein effectuated has not been overcome because the above arguments are mostly aimed at a previous Office ground of rejection, which no longer applies as per the latest amendments to the Claims.

In all, the claims stand rejected as set forth in the Office Action.

Conclusion

11. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Art Unit: 2193

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

June 15, 2008